SEC-PROJECT 2019 Stage 2 Report

G6 - 78093, 83420, 84740

We built a spam system using a proof of work (pow) system called HashCash[1]. This system is used to limit email spam by forcing the email sender to expend a modest amount of CPU before sending the email. We incorporated this algorithm to generate the proof of work in our utility library, and we use it in the notary client. Before each request the Notaries oblige that a proof of work with 20 partial preimage zero bits is performed. The Notaries also enforce that the resource string in the proof of work must be the sha-1 sum of the concatenation of the entire request contents. This approach guarantees this pow was generated for this particular request and that an attacker does not have pre-calculated pow's to immediate use.
Once the Notaries receive a request, they start by verifying the leading zeros of the HashCash sha-1 hash, if the resource string matches the received parameters, and validity of the pow (the day it was generated).

The system is now composed by N replicas of the Notary, where N is equal to 3*F+1 and F corresponds to a maximum number of faulty servers which is defined as a static variable in the Notarys and in the client library. In order to have a (1,N) Byzantine Regular register in the system, we implemented the Fail-Arbitrary Algorithm: Authenticated-Data Byzantine Quorum. So, when a user does a write operation, the user sends in addition to all parameters already sented in the previous project stage, a write signature which is composed by the actual or future ownerID, the goodID, the timestamp and a boolean defining whether it is on Sale. The write operation is sent to all replicas and the user waits for N+F/2 ACKS, in order to complete a Byzantine quorum. The replicas upon receiving the write operation, in addition to the usual verifications, they also check if the received write signature is correct. If everything is ok, they perform the write operation. When a user does a read operation, he sends the request to all replicas and awaits for (N+F)/2 answers with correct write signature, completing a byzantine quorum. After receiving the demanded answers, the user verifies which answer has the highest timestamp and returns it.

In order to transform the system in a (1,N) Byzantine Atomic register, we added a write back phase to the read performed in the users. So basically, whenever a user makes a read operation, after receiving the correct answer, the user sends a write to all replicas with that answer and awaits for (N+F)/2 ACKS. This ensures that at least there is a byzantine quorum with the most recent value, preventing from returning the previous written value.

After implementing this two protocols, the system was still vulnerable to byzantine clients. Since, the clients could send a write operation only to one replica and be accepted, or send a write operation with different values to different replicas and also be accepted. So it was implemented the Authenticated Double Echo Broadcast, to mitigate those kind of attacks that could compromise the system integrity. So, whenever a notary receives a write operation, it broadcasts an echo to all replicas, including himself, with the write values received and awaits for other echos. When the notary receives (N+F/2) echo messages from the other replicas with the same write value, it broadcasts a ready message to all replicas including himself and awaits for 2*F+1 ready answers with the same write value. If the

[1] "Hashcash - A Denial of Service Counter-Measure." http://www.hashcash.org/hashcash.pdf.

replica has received F+1 ready with the same write values, it will also broadcast the ready message. In the case of a replica not receiving the write operation or it being delayed, the replica will just add the received echos and writes until it has the necessary amount of answers needed to determine if it is a valid write. The notaries must receive the readys and echos necessary within a session time of 5 sec. If they don't receive the necessary answers, the write operation is considered invalid. If the write operation is considered valid, the notary verifies if it has received the previous valid write operation, and if it didn't received, it waits for a session time for the previous write operation or until it arrives it. The actual or future ownerID together with the timestamp are used to identify the write operation. Whenever a Notary receives a message, it checks the received signatures to prevent attacks.
In order to this protocols to work, we make sure that our system is composed by AuthPerfectPointToPointLinks.